Lab 9: Input and Output in RARS

Taaruni Ananya | Session 008 | CRN: 15623

03/28/2025

Dr. Lan Gao

Introduction

Lab 9 is centered around programming for taking input and returning an output, which involves practicing writing code with RARS, SNOWBALL, and Terminal as well as learning I/O redirection and piping. Lab 9 looks into exploring how to use RARS to write a program that can take in input, perform an action, and return an output. The lab explains how to use terminal for RARS, jar, and testing with SNOWBALL. The lab also gives hands-on experience with writing a program to perform these actions.

Purpose

The purpose of this lab is to learn how to program efficiently as well as the use of STDIN and STDOUT, along with the commands and practice writing RARS utilizing Terminal; this lab also works to reintroduce commands like java -jar, vi, cat, and rm. This lab is meant to experiment with using terminal to take in input and return output with programs and explore different ways to use these commands.

Main Objectives

- Learn how to read from STDIN
- Learn how to write to STDOUT
- Practice the li, la, sw, .data, .eqv, str, int, char, mybuffer, ecall, sb, beq, and j–as well as eof_reached
- Learn how to test a program with piped input, input redirection, and use output redirection
- Practice using Terminal to write and test code
- Practice RARS

Brief Explanation

RARS stands for RISC-V Assembler and Runtime Simulator, a software tool used for writing, assembling, and simulating RISC-V assembly language programs. “Terminal is an app for advanced users and developers that lets you communicate with the Mac operating system using a command line interface (CLI).” (Apple). Using Terminal, a programmer can use RARS to take in inputs, perform operations, and return outputs. Terminal provides a place to run programs that can output errors, what’s causing them, what line, and which column to better assist programmers.

Methodology/Procedure

The procedure starts with writing a program to take in input for char, int, and a string, then returning the number of chars read. This program uses STDIN to read from, STDOUT to write to, using an array ro reserve “BUFFER_SIZE” bytes of space at the label "mybuffer". Reading a charr, int, and string, finding the number of chars read, and returning it. Next, the same program is continued with code to space, ASCII characters like “!”, and improve the first part’s code. Finally, part 3 uses code from lab9_pt2.s to address the problems such as using the read with an fd (READ) which does not work when input is redirected, using the read string (SIMPLE_READ), and the newline character that’s used even when the string is empty.

Documentation

```
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar lab9_pt1.s
RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter a char 2
Enter an int 13
Enter a string abc
Number of chars read is 4
Enter a string
Number of chars read is 1
Enter a string
```

```
.eqv  BUFFER_SIZE  10

.align 2
    mybuffer: .space   BUFFER_SIZE
```

```
.eqv  STDIN        0
    .eqv  STDOUT       1
    .eqv  STDERR       2
    .eqv  READ         63
    .eqv  WRITE        64
    .eqv  SIMPLE_READ  8
    .eqv  SIMPLE_WRITE 4
    .eqv  NL           10
```

```
# Read a character
    li      a7, 12    # ecall code for read character
    ecall
```

```
# Read an integer
    li a7, 5
    ecall
```

```
# Read a string
    li  a7, READ        # read a string from file given
    by fd
    li  a0, 0           # fd is 0
    la  a1, mybuffer
    li  a2, BUFFER_SIZE
    ecall
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar testing.s
RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

Enter a char 2
Enter an int 13
Enter a string abc
Number of chars read is 4
Enter a string Number of chars read is 0
String read is !abc
!
Int value read is !13!
Char value read is !50!
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ vi lab9_pt3.s
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar lab9_pt3.s < testfile.txt
RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

String read !pear
!
String read !duck
!
String read !hop
!
Char value read !w!
Int value read !82!
```

Questions:

Part One:

1. **Why are "Enter a string" and "Number of chars read is 0" on the same line?**

   The “Enter a string” and “Number of chars read is 0” are on the same line because of the newline character’s use in this program.In this case, the user does not enter a value,

resulting in the “Number of chars read is 0” being activated as a result and does not print on a new line.

2. **Suppose that the user does not follow directions, and enters "1" for "Enter a char", then "a" for "Enter an int". What will happen?**

   Sine this program uses ecall, an user entering differing values to what’s asked like entering “1” for “Enter a char” or “a” for “Enter an int” would cause an error to occur. This happens because the program is written in the way to expect a certain type of input. For example, “Enter a char” would be expecting a character, or “Enter an int” is expecting an integer value. Getting a different type than what’s expected would cause issues and an error to pop up.

3. **Did you (or the given code) need to do anything to make sure that the register holding the length is not over-written?**

   The given code is written efficiently enough to make sure that the register is holding the length and is not overwritten. No changes were made on this front because the value is stored in a register.

Part Two:

1. **Why are the exclamation marks on different lines?**

   There are exclamation marks on different lines because they are printed after the string, integer, or character. This may be happening because of the newline character being utilized–which may take the exclamation mark as something that needs to be printed after.

```
String read is !abc
!
Int value read is !13!
```

2. **What happens if the user enters several strings before entering CTRL-D?**

   Since CTRL-D is triggering EOF, the program will keep running until this command is entered. In depth, the program will keep reading input, performing an action, and outputting results over and over again until CTRL-D triggers EOF.

```
# if we read 0 chars, we reached EOF
beq     a0, x0, eof_reached
j       read_again
```

3. **Why is the char value reported as 50?**

   The char value for the given input of "2" is reported as 50 because of ASCII. According to Geeks for Geeks, the character input can be read as a byte of ASCII value, hence why the char value is reported as 50.

4. **What happens if the user types CTRL-D (without entering anything else) the first time that the program asks for a string?**

   If the user types CTRL-D without entering anything else the first time that the program asks for a string, this would trigger EOF and skip all the other prompts (int, char, etc.) to print whatever the program has stored.

```
# if we read 0 chars, we reached EOF
beq     a0, x0, eof_reached
j       read_again
```

5. **What happens if the user types CTRL-D (after entering "abc") the first time that the program asks for a string?**

   If the user types CTRL-D after entering “abc” the first time that the program asks for a string, the program takes this input, finds the number of chars, and returns “number of chars read 4”, stopping all the other prompts and returning this value.

```
Enter a char 2
Enter an int 13
Enter a string abc
Number of chars read is 4
Enter a string Number of chars read is 0
```

6. **If you enter a string that is larger than the buffer, how many chars read does the program report?**

   In this program, the buffer size is ten, given by BUFFER_SIZE = 10. If more than characters are entered, larger than the buffer, reads only up to ten chars and returns this value because it is only capable of holding this many values.

```
.eqv  BUFFER_SIZE,  10
```

7. **If you enter a string that is larger than the buffer, then enter a string that is shorter and press CTRL-D, what does the program report as the string read, and why?**

   If a user enters a string that is larger than the buffer, then enters a string that is shorter, and presses CTRL-D, the program reports the last number of characters read that align with the program's limitations as the string reads.

Part Three:

1. **Do we get the same results whether the input comes from "echo", "cat", or input redirection ("<")?**

   If the same testfile.txt with the same inputs are used to call the program with all three: "echo "s\n42\ncat" | java -jar rars1_6.jar lab9_pt3.s", "cat testfile.txt | java -jar /home/mweeks/rars1_6.jar lab9_pt3.s", and "java -jar /home/mweeks/rars1_6.jar lab9_pt3.s < testfile.txt", they will yield the same results.

2. **Do we get the same results whether the output goes to the terminal or a file using output redirection (">")?**

   Yes, after testing both types of code with "java -jar /home/mweeks/rars1_6.jar lab9_pt3.s < testfile.txt", and "java -jar /home/mweeks/rars1_6.jar lab9_pt3.s > testout", the same results were yielded each time.

3. **What is the advantage of using input redirection and output redirection versus entering it interactively?**

   There are a few advantages to using input redirection and output redirection versus entering it interactively such as efficiency and pipelining. Using input redirection and output redirection reduces the number of times an input needs to be entered, for example. As for pipelining, commands can be completed–this helps with efficiency as well.

Key Code Observations

```
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar lab9_pt1.s
RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar
```

```
Enter a char 2
Enter an int 13
Enter a string abc
Number of chars read is 4
Enter a string
Number of chars read is 1
Enter a string
```

```
.eqv  BUFFER_SIZE  10

.align 2
    mybuffer: .space   BUFFER_SIZE
```

```
.eqv  STDIN        0
    .eqv  STDOUT       1
    .eqv  STDERR       2
    .eqv  READ         63
    .eqv  WRITE        64
    .eqv  SIMPLE_READ  8
    .eqv  SIMPLE_WRITE 4
    .eqv  NL           10
```

```
# Read a character
    li      a7, 12    # ecall code for read character
    ecall
```

```
# Read an integer
    li a7, 5
    ecall
```

```
# Read a string
    li  a7, READ         # read a string from file given
    by fd
    li  a0, 0            # fd is 0
    la  a1, mybuffer
    li  a2, BUFFER_SIZE
    ecall
```

```
Enter a char 2
Enter an int 13
Enter a string abc
Number of chars read is 4
Enter a string Number of chars read is 0
String read is !abc
!
Int value read is !13!
Char value read is !50!
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ vi lab9_pt3.s
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar lab9_pt3.s < testfile.txt
RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

String read !pear
!
String read !duck
!
String read !hop
!
Char value read !w!
Int value read !82!
```

```
# if we read 0 chars, we reached EOF
beq     a0, x0, eof_reached
j       read_again
```

```
# write a string
    li  a7, WRITE       # write to file
    li  a0, STDOUT      # fd for STDOUT
    la  a1, mybuffer
    la  t0, temp
    lw  a2, 0(t0)       # number of chars
    ecall
```

```
# Did we get a NL?
     la      t0, mybuffer
     lb      a0, 0(t0)            # Char to check
     li      a1, NL
     beq     a0, a1, eof_reached  # Did we read 0 chars?
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ echo "s\n42\ncat" |  java -jar rars1_6.jar lab9_pt3.s
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ cat testfile.txt | java -jar /home/tananya1/rars1_6.jar lab9_pt3.s
RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

String read [nea
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar lab9_pt3.s
RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar lab9_pt3.s < testfile.txt
RARS 1.6  Copyright 2003-2019 Pete Sanderson and Kenneth Vollmar

String read [nea
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar lab9_pt3.s > testout
```

```
[tananya1@gsuad.gsu.edu@snowball ~]$ java -jar /home/tananya1/rars1_6.jar lab9_pt3.s < testfile.txt > testout
```